



УНИВЕРЗИТЕТ У НОВОМ САДУ ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Татјана Ерић
Број индекса: РА 207-2013

Тема рада: Развој окружења за тестирање *cloud* сервиса гласовне контроле

Ментор рада: проф. др Иштван Пап

Нови Сад, јул, 2017



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Татјана Ерић		
Ментор, МН:	проф. др Иштван Пап		
Наслов рада, НР:	Развој окружења за тестирање <i>cloud</i> сервиса гласовне контроле		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2017		
Издавач, ИЗ:	Ауторски репринг		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/слика/графика/прилога)	7/30/9/5/18		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	тестирање, гласовна команда, паметна кућа		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	Тестирање је веома битна ставка сваког развојног пројекта. Намена тестирања је да покаже да програмска подршка ради оно за шта је намењена и да утврди дефекте пре него што се стави корисницима на располагање. У раду је описан развој окружења као и начини тестирања <i>cloud</i> сервиса гласовне контроле.		
Датум прихватања теме, ДП:			
Датум одбране, ДО:			
Чланови комисије, КО:	Председник:	доц. др Јелена Ковачевић	
	Члан:	доц. др Богдан Павковић	Потпис ментора
	Члан, ментор:	проф. др Иштван Пап	



KEY WORDS DOCUMENTATION

Accession number, ANO:			
Identification number, INO:			
Document type, DT:	Monographic publication		
Type of record, TR:	Textual printed material		
Contents code, CC:	Bachelor Thesis		
Author, AU:	Tatjana Erić		
Mentor, MN:	Ištván Papp, Phd		
Title, TI:	The development environment for testing cloud service based on voice control		
Language of text, LT:	Serbian		
Language of abstract, LA:	Serbian		
Country of publication, CP:	Republic of Serbia		
Locality of publication, LP:	Vojvodina		
Publication year, PY:	2017		
Publisher, PB:	Author's reprint		
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6		
Physical description, PD: <small>(chapters/pages/ref./tables/pictures/graphs/appendices)</small>	7/30/9/5/18		
Scientific field, SF:	Electrical Engineering		
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems		
Subject/Key words, S/KW:	Testing, voice command, smart home		
UC			
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia		
Note, N:			
Abstract, AB:	Testing is an important thing in any development project. The purpose of testing is to show that the application does what it is intended for and to determine the defect of the application before it is put into use. This paper describes the development of the environment and methods to test the cloud service for voice control.		
Accepted by the Scientific Board on, ASB:			
Defended on, DE:			
Defended Board, DB:	President:	Jelena Kovačević, Phd	
	Member:	Bogdan Pavković, Phd	Menthor's sign
	Member, Mentor:	Ištván Papp, Phd	

Zahvalnost

Zahvaljujem se prof. dr Ištvanu Papp i Mariji Antić na stručnoj pomoći i savetima za izradu ovog rada.

Posebno se zahvaljujem svojoj porodici na pruženoj podršci u toku čitavog školovanja i kolegama iz tima.



УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



SADRŽAJ

1.	Uvod	1
2.	Teorijske osnove.....	3
2.1	Osnove testiranja.....	3
2.2	Oblo sistem	6
2.3	Kontrola glasom u pametnim kućama	7
2.4	JSON format podataka.....	9
2.5	MQTT protokol u sistemu za kontrolu pametne kuće	10
3.	Koncept rešenja	11
3.1	Arhitektura okruženja za testiranje <i>cloud</i> servisa glasovne kontrole.....	11
4.	Programsko rešenje.....	15
4.1	Generisanje JSON datoteka	15
4.2	Prikupljanje informacija sa centralog uređaja.....	16
4.3	Komunikacija sa <i>cloud</i> glasovnim servisom.....	17
4.4	Dobijanje odgovora od <i>cloud</i> glasovnog servisa i testiranje	17
4.4.1	Proces kreiranja očekivanog odgovora.....	18
5.	Rezultati testiranja	19
5.1	Rezultati testiranja komadni Dim servisa	19
5.2	Rezultati testiranja komadi Light servisa.....	20
5.3	Rezultati testiranja komandi senzor servisa	22
6.	Zaključak	24
7.	Literatura	25

SPISAK SLIKA

Slika 2.1.1 Proces testiranja	4
Slika 2.1.2 Funkcionalno testiranja (Black box).....	4
Slike 2.1.3 Nefunkcionalno testiranje (White box)	5
Slika 2.2.1 Komunikacija između klijentske aplikacije i uređaja unutar lokalne mreže ..	6
Slika 2.2.2 Komunikacija između klijentske aplikacije i uređaja van lokalne mreže	7
Slika 2.3.1 Tok komunikacije između Amazon glasovnog servisa i Oblo <i>clouda</i>	8
Slika 2.4.1 Primer JSON datoteke za testiranje <i>cloud</i> sistema glasovne kontrole.....	9
Slika 3.1.1 Globalno funkcionisanje okruženja za testiranje.....	12
Slika 3.1.2 Sistem funkcionisanja testiranja u okruženju	13
Slika 4.1.1 Funkcija za generisanje JSON datoteka	15
Slika 4.2.1 Funkcija za prikupljanje uređaja.....	16
Sliku 4.2.2 Funkcija za učitavanje iz fajla	17
Slika 4.3.1 Komunikacija sa <i>cloud</i> servisom glasovne kontrole	17
Slika 4.4.1 Funkcija za poređenje očekivanog i dobijenog odgovora	18
Slika 5.1.1 Pozitivni i negativni testni slučajevi komandi Dim servisa.....	20
Slika 5.2.1 Pozitivni i negativni testni slučajevi komandi Light servisa	21
Slika 5.2.2 Neuspešno izvršen testni slučaj kod Light servisa	22
Slika 5.3.1 Pozitivni i negativni testni slučajevi komandi namenjenih senzorima.....	23

SPISAK TABELA

Tabela 2.1.1 Osnove smernice prilikom testiranja	6
Tabela 3.1.1 Skup servisa unutar okruženja za testiranje.....	13
Tabela 5.1.1 Skup komandi Dim servisa	20
Tabela 5.2.1 Skup komandi Light servisa.....	21
Tabela 5.3.1 Skup komandi senzor servisa.....	22

SKRAĆENICE

MQTT - Protokol za razmenu podataka po principu objave/preplate (eng. Message Queuing Telemetry Transport)

JSON - format za razmenu podataka (eng. JavaScript Object Notation)

AWS - eng. Amazon web service

IP – eng. Internet protocol

1. Uvod

Popularnost rešenja za automatizaciju pametne kuće značajno je porasla poslednjih godina, a naročita pažnja je u poslednje vreme posvećena kontroli pametne kuće glasom. Do nedavno dostupna rešenja za automatizaciju pametnih kuća, bila su zasnovana na korišćenju mobilnih i internet aplikacija. Međutim, uređaji povezani sa sistemima za prepoznavanje govora su postali veoma popularni poslednjih godina, a osim zabave koju pružaju, oni omogućavaju i kontrolu pametnih kuća glasom. Kontrola glasom u pametnim kućama značajno olakšava svakodnevne aktivnosti čoveka, ali njena implementacija predstavlja veoma složen proces.

Testiranje predstavlja veoma bitnu stavku svakog razvojnog projekta. Svi korisnici očekuju ispravnost rada programske podrške. Stoga je, pre stavljanja programske podrške krajnjim korisnicima na raspolaganje, neophodno izvršiti testiranje kako bi se proverila njena funkcionalost i ispravile eventualne greške. U ovom radu je opisan razvoj okruženja za testiranje *cloud* servisa glasovne kontrole, implementiranog u okviru Oblo sistema za kućnu automatizaciju koji koristi Amazon Aleksu kao servis za prepoznavanje govora.

Ručno testiranje implementiranog modula za glasovnu kontrolu sistema podrazumevalo bi svakodnevno ponavljanje različitih pojedinačnih komandi. Na osnovu dobijenih rezultata, moglo bi se ustanoviti postojeće greške, samo bi sam proces testiranja zahtevao veoma dugačak vremenski period. Da bi se izbegao takav način testiranja softvera, razvijeno je okruženje za automatsko testiranje implementiranog *cloud* servisa glasovne kontrole.

Okuženje za testiranje podeljeno je na više različitih modula, a svaki od tih modula namenjen je testiranju komandi za određeni servis koji uređaji podržavaju. Cilj ovog rada je da se izvrši detaljno testiranje svakog servisa posebno i da se analiziraju rezultati kako bi se

ispravile eventualne greške pre puštanja same programske podrške krajnjim korisnicima na raspolaganje.

Rad sadrži sedam poglavlja. U drugom poglavlju su opisane teorijske osnove koje su neophodne za razvoj okruženja za testiranje *cloud* servisa glasovne kontrole. Treće poglavlje sadrži koncept rešenja za izradu okruženja za testiranje. Unutar četvrtog poglavlja detaljno je opisana implementacija okruženja za testiranje *cloud* servisa glasovne kontrole.

U petom poglavlju su predstavljeni rezultati testnih slučajeva. Poslednja dva poglavlja predstavljaju zaključak i literaturu koja je korišćena tokom izrade ovog rada.

2. Teorijske osnove

2.1 Osnove testiranja

Testiranje predstavlja proces u razvoju programske podrške, tokom koga je cilj pronaći greške u programskoj podršci koja se razvija. Svako ponašanje programske podrške koje se ne slaže sa originalnim zahtevima predstavlja grešku, propust ili neispunjeni zahtev koji je potrebno identifikovati i otkloniti. Namena testiranja je da pokaže da programska podrška radi ono za šta je namenjena, kao i da se utvrde i isprave defekti programske podrške pre nego što se stavi krajnjim korisnicima na raspolaganje. Prilikom testiranja programske podrške koriste se unapred definisani podaci, a rezultati testova treba da pokažu greške ili ukažu na nepredviđeno ponašanje programske podrške. Testiranje obuhvata proces validacije i verifikacije. Validacija predstavlja proces koji proverava da li programska podrška zadovoljava korisničke zahteve na kraju životnog ciklusa, dok verifikacija predstavlja proces koji proverava da li programska podrška implementira određenu funkcionalnost u skladu sa specifikacijom. Osnovni cilj validacije i verifikacije je uspostavljanje uverenja da sistem u potpunosti ispunjava svoju namenu [1].

Testiranje predstavlja proces koji obuhvata veliki broj aktivnosti. Neke od neophodnih aktivnosti za testiranje svake programske podrške su:

- **Strategija testiranja** - strategija testiranja definiše celokupan pristup testiranju, kao što su nivoi testiranja, metode, tehnike i alati.
- **Plan testiranja** - sadrži spisak test slučajeva, definiše učestanost izvršavanja testova, njihove prioritete, kao i konkretne zadatke koji će omogućiti ostvarenje strategije testiranja.
- **Implementacija testa** - pisanje detaljne procedure testiranja, priprema podataka.

- **Izvršenje testa** - izvršenje test slučajeva i scenarija, izveštaj o greškama i izveštaj testiranja.
- **Analiza rezultata** - proveravanje zašto je došlo do grešaka, sugestije i zapažanja.

Primer aktivnosti testiranja koje se primenjuju tokom faza razvoja projekta (Slika 2.1.1).



Slika 2.1.1 Proces testiranja

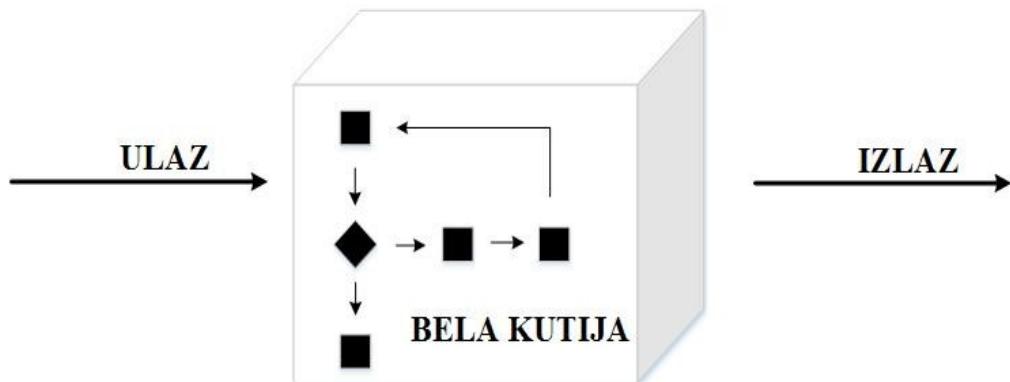
Postoji više načina za testiranje programske podrške. Neki od značajnijih tipova testiranja su funkcionalno i nefunkcionalno testiranje.

1. **Funkcionalno testiranje** (Black-Box) tretira programsku podršku kao zatvorenu kutiju, tj. testiranje ne gleda programski kod, već se samo šalju ulazni podaci. Prati se izlaz i proverava se da li se očekivani rezultati podudaraju sa dobijenim. Cilj ovakvog načina testiranja je pronaći funkcije koje nedostaju ili su neispravne, zatim greške u ponašanju programske podrške ili greške inicijalizacije (Slika 2.1.2).



Slika 2.1.2 Funkcionalno testiranja (Black box)

2. **Nefunkcionalno testiranje** (White-Box) programsku podršku posmatra kao otvorenu kutiju, jer proverava programski kod. Rezultati testiranja se dobijaju ispitivanjem logike programske podrške. Cilj nefunkcionalnog testiranja je pokrivanje grananja programskog koda (testiranje *if, for, while* i drugih petlji) graničnim slučajevima, kao i odvojeno testiranje različitih funkcionalnih celina programske podrške (Slika 2.1.3).



Slike 2.1.3 Nefunkcionalno testiranje (White box)

Neki od osnovnih tipova testova koji se mogu izvršavati su sledeći:

- Jedinično testiranje (eng. *Unit testing*) - testiranje svake atomske jedinice koda.
- Integraciono testiranje (eng. *Integration testing*) - testiranje sistema programske podrške kao celine.
- Testiranje sigurnosti (eng. *Security testing*) - ovom vrstom testiranja utvrđuju se ranjivosti sistema.
- Regresiono testiranje - je vrsta testiranja koja nastoji da otkrije nove greške programske podrške, prouzrokovane promenama tokom unapređivanja funkcionalnosti ili promena konfiguracije.
- Test konfiguracije (eng. *Configuration testing*) - testira ponašanje programske podrške u različitim okruženjima.

Svaka programska podrška prilikom testiranja zahteva sve od gore navedenih tehnika za testiranje, kako bi se na osnovu tih testova dobili rezultati i zaključila ispravnost sistema.

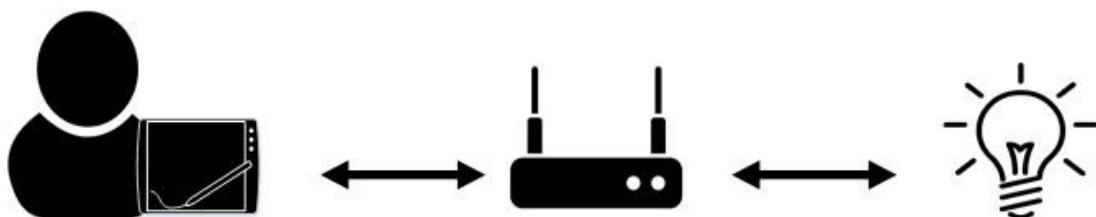
Pre svakog testiranja neophodno je pratiti smernice prikazane u tabeli 2.1.1.

	Smernice prilikom testiranja
1	Prilikom pravljenja testova neophodno je prvo definisati očekivani rezultat.
2	Testove pišemo za neispravne i neočekivane ulazne vrednosti, kao i za ispravne i očekivane ulazne vrednosti.
3	Ne treba pisati testove koji ne mogu da se ažuriraju.
4	Proces testiranja mora da ima opširnu analizu rezultata.
5	Ne planirati testove na pretpostavkama da neće biti grešaka.
6	Programsku podršku nikada ne testira onaj ko ju je pisao.

Tabela 2.1.1 Osnove smernice prilikom testiranja

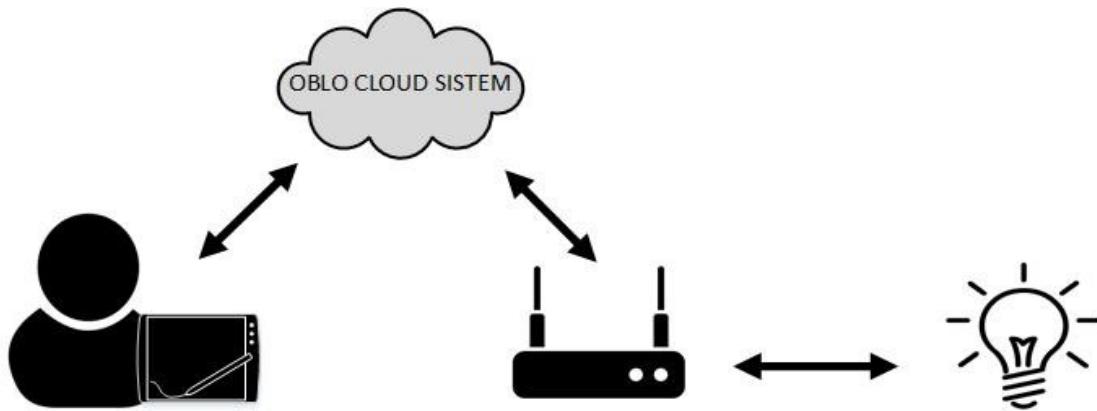
2.2 Oblo sistem

Oblo sistem se sastoji od više nezavisnih celina koje komuniciraju međusobno putem različitih protokola. Sistem koristi *Zigbee* [2] 2.4GHz i *Z-Wave* [3] bežične mreže za efikasno upravljanje i konfiguraciju uređaja. Centralni uređaj (eng. *gateway*) predstavlja sponu između IP (LAN/WiFi) mreže i uređaja, i ima mogućnost da kontroliše sve uređaje u kući. Centralni uređaj obrađuje naredbe koje prihvata od klijenta (korišćenjem mobilne aplikacije), i dobavlja podatke o stanju uređaja. Poruke koje se razmenjuju između centralnog uređaja i klijentske aplikacije su formatirane kao JSON objekti. Veza između centralnog i klijentskih uređaja je bežična ali podrazumeva povezivanje u okviru lokalne mreže posredstvom mrežnih rutera (Slika 2.2.1).



Slika 2.2.1 Komunikacija između klijentske aplikacije i uređaja unutar lokalne mreže

Ukoliko se klijentski uređaj nalazi izvan lokalne mreže, mora postojati drugi način za pristup centralnom uređaju. Za tu namenu uvodi se “okruženje u oblaku” ili *cloud* rešenje, koje predstavlja instancu poslužioca koja je dostupna na javnoj IP adresi, i konstantno održava vezu sa centralnim uređajem. Komunikacija od strane centralnog uređaja, a i klijentske aplikacije prema *cloud* platformi, bazirana je na MQTT protokolu po principu “pretplati se/objavi” (Slika 2.2.2). *Cloud* servis realizovan je pod Node.js platformom [4].



Slika 2.2.2 Komunikacija između klijentske aplikacije i uređaja van lokalne mreže

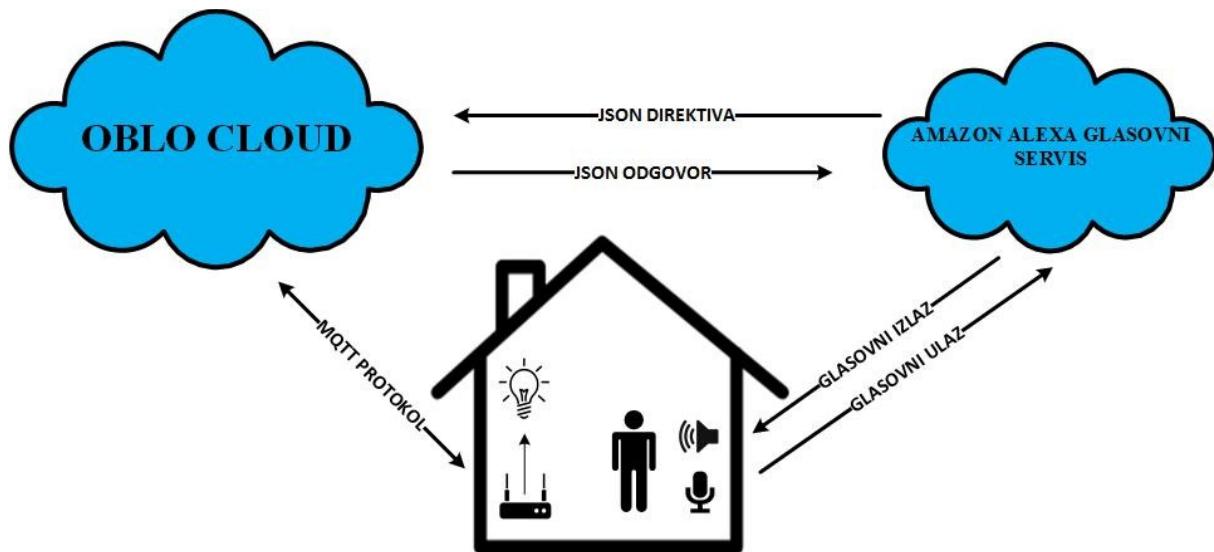
2.3 Kontrola glasom u pametnim kućama

Kontrola glasom u pametnim kućama se najbrže i najjednostavnije može realizovati pomoću nekog od već postojećih sistema za prepoznavanje govora. Za realizaciju modula za glasovnu kontrolu u okviru Oblo sistema kućne automatizacije korišćen je Amazon Aleksa glasovni servis (eng. *Amazon Alexa Voice Service*). Jedna od najvećih prednosti ovog servisa je mogućnost kreiranja aplikacije, koja se poziva prilikom izgovaranja određenih komandi i obezbeđuje željenu funkcionalnost prilagođenu Oblo sistemu. Da bi se napravila aplikacija za Aleksa servis najpre je neophodno napraviti Amazon AWS nalog [5], a zatim je potrebno zadati ime i naziv aplikacije, kao i okidač (ključnu reč) kojim se ona pokreće. Dalje, pri konfiguraciji aplikacije potrebno je definisati korisničke namere (eng. *user intents*), tj. obrasce komandi koje se mapiraju na određenu funkcionalnost u okviru Oblo *cloud* servisa. Sama funkcionalnost implementirana je u vidu metoda u okviru Oblo *cloud* servisa. Svaka aplikacija mora imati ime (eng. *invocation name*) koje se pri pozivu mora izgovoriti. Namena i modeli (eng. *utterance*) moraju biti predefinisani kako bi Amazon prepoznavanje govora bilo precizno jer mu se na taj način zadaje šta može očekivati na ulazu. Ukoliko se ne zadaju, prepoznavanje je moguće, ali je dosta otežano i sistem radi stohastički. Ova osobina je ujedno

i jedna od najvećih nedostataka aplikacije jer je korisničko ponašanje često nepredvidivo, te nije uvek moguće nabrojati sve moguće reči, fraze i kombinacije [6]. Celokupno prepoznavanje govora se odvija u okviru Aleksa *cloud-a* čime dobijamo tekstualni format izrečene naredbe u vidu JSON objekta.

U okviru ove aplikacije se definiše celokupna logika i ostvaruje se komunikacija kako sa Amazon glasovnim servisom, tako i sa OBLO sistemom. Cilj aplikacije jeste da, na zadatu glasovnu komandu koju Amazon glasovni servis prepozna, spakuje u odgovarajući JSON paket i šalje servisu implementiranom na OBLO *cloud-u*, u kojem se najpre parsira primljeni JSON, a zatim mapira zahtev (eng. *request*) i željeni uređaj (eng. *device*) koji je naveden u komandi na postojeće uređaje i zahteve vezane za njih.

Nakon toga OBLO *cloud*, komunicira sa centralnim uređajem koristeći MQTT protokol, a zatim, centralni uređaj šalje komandu odgovarajućem uređaju koji je izvršava (Slika 2.3.1).



Slika 2.3.1 Tok komunikacije između Amazon glasovnog servisa i Oblo *clouda*

Aplikacija šalje odgovor o uspešnosti izvršavanja akcije posredstvom uređaja za prepoznavanje glasa.

2.4 JSON format podataka

Podaci se između Oblo *cloud* i Amazon Aleksa servisa razmenjuju u **JSON** formatu [7]. JSON format podataka koristi tekstualne datoteke za prenos objekata koji poseduju sledeće dve strukture :

- Zbirka parova ime-vrednost – Može se koristiti za prenos objekata realizovanih kao zapis, struktura, lista sa ključevima, heš tabela, asocijativni niz.
- Uređena lista vrednosti – Koristi se za prenos objekata realizovanih kao niz, vektor, lista ili sekvenca.

Različiti tipovi podataka neophodnih za izvršavanje komande se smeštaju u JSON datoteku, o čemu će biti reči u sledećem poglavlju. Primer JSON datoteke kojim se vrši paljenje lampe u okviru Oblo sistema kućne automatizacije dat je na slici 2.4.1.

```
{
  "request": {
    "intent": {
      "name": "LightIntent",
      "slots": {
        "LightRequests": {
          "name": "LightRequests",
          "value": "turn on"
        },
        "Things": {
          "name": "Things",
          "value": "lamp"
        }
      }
    },
    "version": "1.0"
  }
}
```

Slika 2.4.1 Primer JSON datoteke za testiranje *cloud* sistema glasovne kontrole

Parsiranje je proces kreiranja JavaScript objekta od JSON tekstualnog podatka. Razlike između JSON strukture i objekata kod JavaScript-a su:

- **Ime** kod JSON strukture može biti bilo koji validan niz karaktera **uključujući i razmake** dok kod JavaScript-a to nije dozvoljeno.
- **Ime** kod JSON strukture mora da bude **u dvostrukim navodnicima** dok se kod JavaScript-a navodnici ne koriste.

Na osnovu prethodno navedenog, neophodno je prebaciti JSON objekat u JavaScript objekat, tj. izvršiti **parsiranje** koje će ukloniti znake navodnika i razmake ako postoje.

2.5 MQTT protokol u sistemu za kontrolu pametne kuće

MQTT (eng. *Message Queuing Telemetry Transport*) je pojednostavljeni protokol za komunikaciju, baziran na principu objavi – pretplati (*publish/subscribe*). MQTT protokol [8], sadrži dva osnovna pojma: „tema“ (eng. *topic*) i „sadržaj“ (eng. *payload*) poruke. Kada su različiti učesnici u komunikaciji zainteresovani za određenu **temu**, oni će se na to pretplatiti, ili objaviti određeni **sadržaj**. Po MQTT protokolu, **broker** je centralna komponenta mreže, odnosno poslužilac u komunikaciji, a **klijenti** su korisnici specifičnih usluga brokera. Sve poruke dolaze do brokera koji služi da ih pri pristizanju preusmeri onim klijentima koji su na temu te poruke pretplaćeni. Na taj način je razdvojeno slanje poruka od njihove isporuke. Slaganjem poruka u redove za isporuku, broker olakšava komunikaciju između dve strane. Komunikacija između Oblo *cloud-a* i centralnog uređaja obavlja se isključivo korišćenjem MQTT protokola.

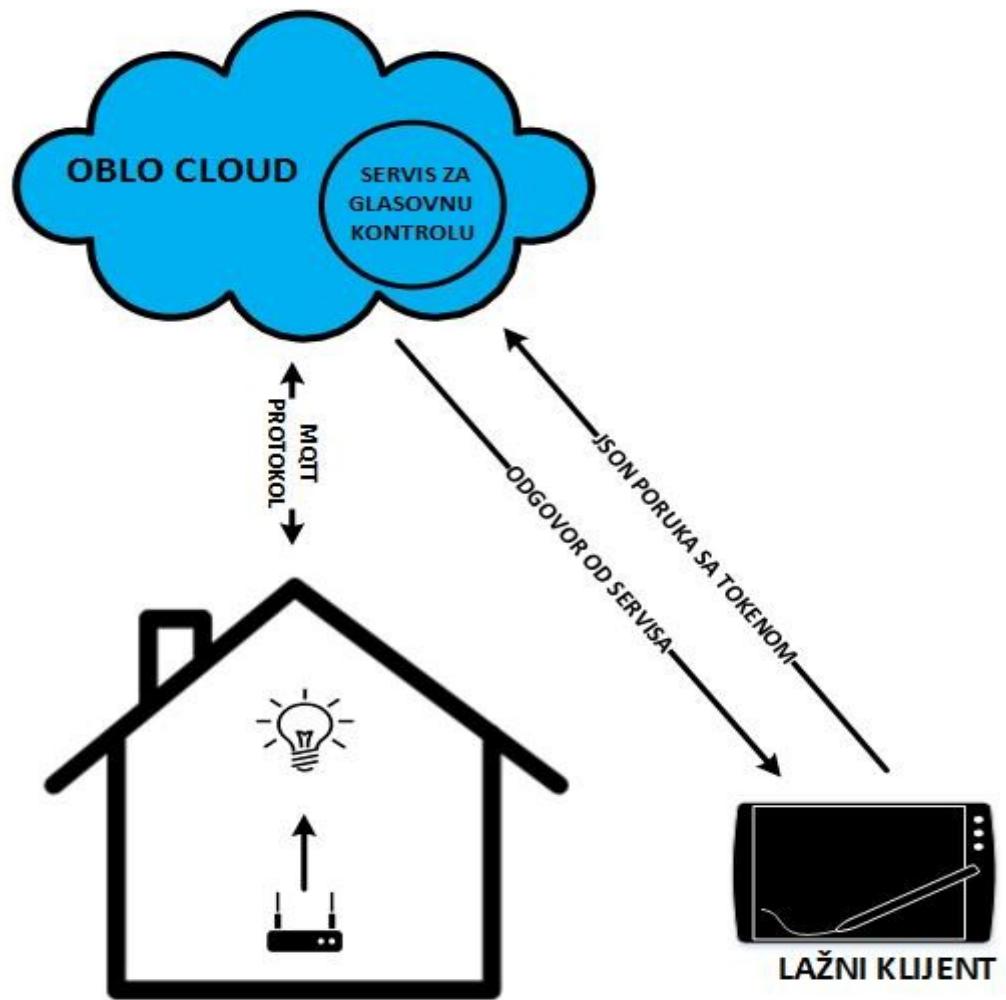
3. Koncept rešenja

Implementirani modul za kontrolu glasovnim komandama realizovan je tako da korisniku potvrđuje izvršenje komande, ukoliko je komanda ispravno formulisana, sadrži sve neophodne parametre, a željeni uređaj se nalazi u korisnikovom sistemu. Takođe, ukoliko u komandi nedostaju neki parametri neophodni za njeno uspešno izvršavanje, ili se željena akcija ne može izvršiti na određenom uređaju, korisnik će o tome biti obavešten. Dodatno, prilikom testiranja je neophodno pokriti i slučajeve usled greške unutar Amazonovog servisa, te format ulazne poruke ne odgovara dogovorenom.

Ideja prilikom razvoja okruženja za testiranje *cloud* sistema glasovne kontrole, bila je da se izvršavanje komandi testira automatski uzastopnim slanjem zahteva glasovnom servisu i poređenjem očekivanog i dobijenog odgovora. Okruženje se zasniva na “*lažnom klijentu*” koji generiše JSON datoteke i šalje ih kao komandu Oblo *cloud* servisu za glasovnu kontrolu pametne kuće.

3.1 Arhitektura okruženja za testiranje *cloud* servisa glasovne kontrole

Na slici 3.1.1, predstavljena je arhitektura okruženja za testiranje *cloud* servisa glasovne kontrole, koja je bazirana na sistemu da “*lažni klijent*” komunicira sa Oblo *Cloud* servisom.



Slika 3.1.1 Globalno funkcionisanje okruženja za testiranje

“Lažni klijent” se sastoji od tri modula namenjena testiranju komandi prilagođenih različitim tipovima uređaja:

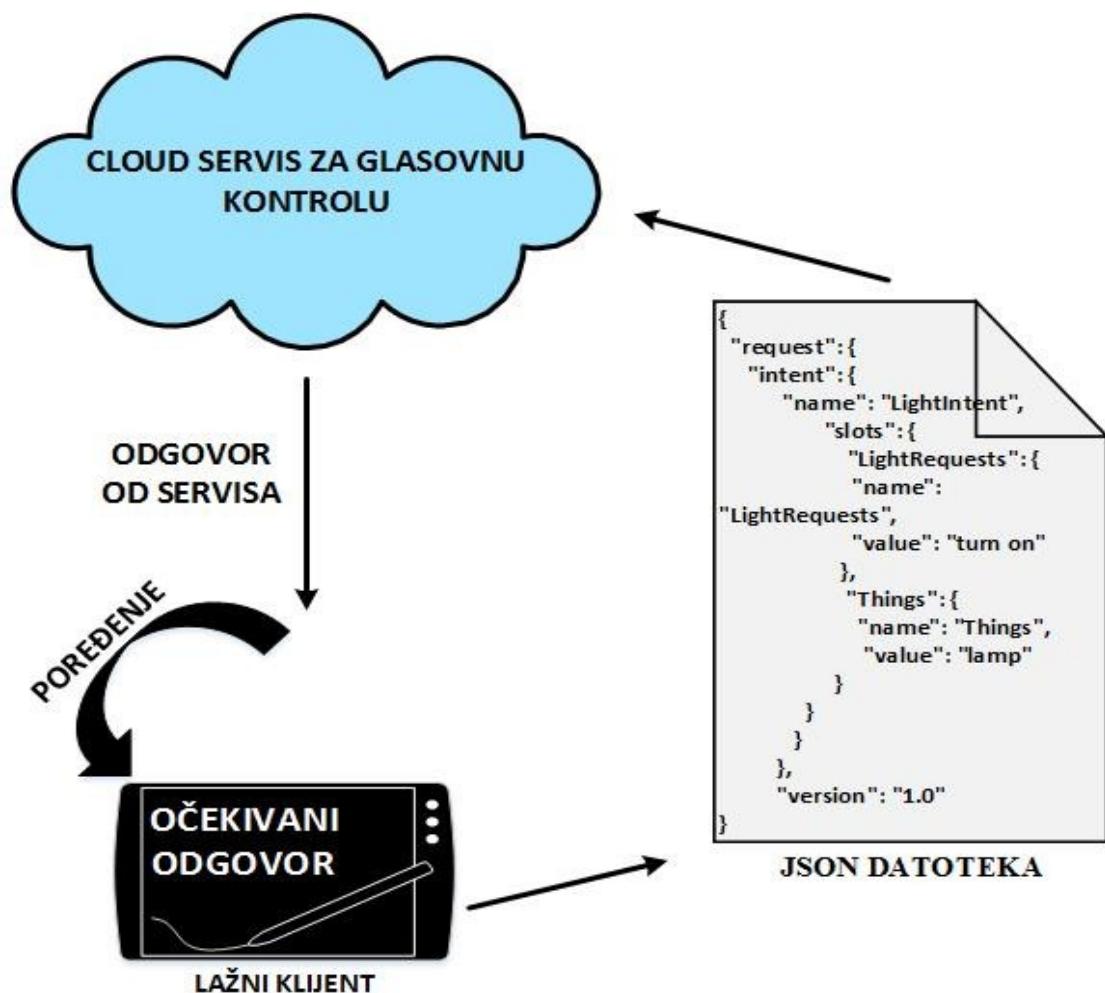
- I. **DimIntent** modul, predstavlja modul za testiranje komandi namenjenih kontroli uređaja sa *Dim* servisom. Dim servis omogućava podešavanje nivoa osvetljenja na željenu vrednost, za uređaje poput prekidača sa potenciometrom, pametnih sijalica i slično.
- II. **LightIntent** modul testira funkcionalnost komandi namenjenih *Light* i *Outlet* servisu. Ove servis podržavaju uređaji kao što su pametne sijalice, prekidači sa potenciometrom, pametne utičnice. Kontrolom parametara ovih servisa omogućava se paljenje i gašenje uređaja, kao i menjanje boja u zavisnosti od funkcije uređaja.
- III. **SensorIntent** modul namenjen je testiranju komandi za proveru stanja različitih senzora. Omogućena je provera stanja senzora za merenje temperature, detekciju poplave, požara i slično.

U tabeli 3.1.1, prikazan je skup različitih servisa i njihovih funkcija.

Broj	Naziv servisa	Naziv funkcije
1.	Dim servis	Upravljanje vrednostima novoga osvetljenja (postavljanje na određenu vrednost osvetljenje uređaja)
2.	Light servis	Paljenje, gašenje i menjanje boja uređaja
3.	Senzor servisi	Proveravanje stanja senzora

Tabela 3.1.1 Skup servisa unutar okruženja za testiranje

Okruženje za testiranje formira komande namenjene modulu za glasovnu kontrolu, u vidu unapred generisanih JSON datoteka. Tako formirane komande se šalju sa "lažnog klijenta". Generisane komande pokrivaju kako pozitivne, tako i negativne test slučajeve. Slikom 3.1.2, opširnije je opisan sam sistem funkcionisanja okruženja za testiranje.



Slika 3.1.2 Sistem funkcionisanja testiranja u okruženju

Proces izvršavanja pojedinačnog testnog slučaja je sledeći:

1. Lažni klijent generiše JSON datoteku i šalje ga *cloud* servisu za glasovnu kontrolu.
2. *Cloud* servis za glasovnu kontrolu obrađuje tu JSON datoteku i vraća nazad odgovor.
3. Odgovor od *cloud*-a se poredi sa unapred definisanim očekivanim odgovorom.

4. Programsко rešenje

Okruženje za testiranje *cloud* servisa glasovne kontrole, razvijeno je na Node.js platformi [9]. Programski jezik koji je korišćen za realizaciju ovog okruženja je JavaScript.

4.1 Generisanje JSON datoteka

Generisanje JSON datoteka izvršeno je pomoću funkcije *formTestInput* koja vraća formirane JSON datoteke koristeći funkciju *formJson*. Funkcija *formJson* vrši dodavanje podataka u izvornu praznu JSON datoteku (Slika 4.1.1.). Argument *obj* je objekat koji sadrži unapred definisane podatke koje se prosleđuju pomenutoj datoteci.

```

function formJson(input, json, obj) {
    json.slots.DimRequests.value = obj.request;
    json.slots.Things.value = obj.thing;
    json.slots.Percentage.value = obj.percentage;
    input.request.intent = json;
    return input;
}

function formTestInput(intentJson, obj, callback) {
    var jsonArray = [];
    var k = 0;
    for(var i = 0; i < obj.length; i++) {
        var input = JSON.parse(JSON.stringify(intentJson));
        var json = JSON.parse(JSON.stringify(obj[i]));
        jsonArray[k] = formJson(input, json, obj[i]);
        ++k;
    }
    callback(jsonArray);
}

exports.formTestInput = formTestInput;

```

Slika 4.1.1 Funkcija za generisanje JSON datoteka

4.2 Prikupljanje informacija sa centralog uređaja

Da bi izvršili testiranje, neophodno je da se pre komunikacije sa *cloud* glasovnim servisom prikupe informacije o perifernim uređajima koji su povezani sa centralnim uređajem, kako bi se definisao očekivani odgovor. Naime, očekivani odgovor na izgovorenu komandu zavisi od toga da li uređaj određenog imena postoji u sistemu ili ne. Za prikupljanje informacija koristi se metoda *getDevices* (Slika 4.2.1).

```
function getDevices(callback) {
    var client = mqtt.connect(options.HOST, options.OPTIONS);
    client.on('connect', (function(connack) {
        client.subscribe(options.subTopic, function() {
            client.subscribe(options.pubTopic, function() {
                client.publish(options.pubTopic, payload.getDeviceListString());
            });
        });
    )));
    client.on('message', (function(topic, message) {
        switch (obj.uid) {
            case GET_DEVICES_UID:
                devProcessing.getAllDevices(obj, function(devs) {
                    for(var i=0; i<devs.length; i++){
                        devices0blo.devices[i] = devs[i].dev_name;
                    }
                    callback(devices0blo);
                });
                fs.appendFileSync('AllDevice.json', JSON.stringify(devices0blo));
                client.end();
            }
        break;
    });
}));
};
```

Slika 4.2.1 Funkcija za prikupljanje uređaja

Lista uređaja se smešta u tekstualnu datoteku, koristeći funkciju *fs.appendFileSync*. Kada se pokrene testiranje, lista uređaja se očitava pomoću funkcije *fileread* (Slika 4.2.2) i omogućuje definisanje validnog očekivanog odgovora.

```

function fileread(filename){
    var contents= JSON.parse(fs.readFileSync(filename));
    return contents;
}

```

Sliku 4.2.2 Funkcija za učitavanje iz fajla

4.3 Komunikacija sa *cloud* glasovnim servisom

Komunikacija se obavlja koristeći funkciju *executeTest* (Slika 4.3.1). Formirane JSON datoteke se šalju na *cloud* svakih 4s sto je definisano funkcijom *setTimeout* (Slika 4.4.1), pri čemu se koristi POST metodu.

```

function executeTest() {

    var post_options = {
        url: 'https://aisha-test.rt-rk.com/vci/alexa',
        method: 'POST',
        json: true,
        body: 'hi'
    };
    formJson.formTestInput(dimJSON, obj, function(jsonObj) {
        for(var i = 0; i < jsonObj.length; i++) {
            var myBody = clone(jsonObj[i]);
            var options = JSON.parse(JSON.stringify(post_options));
            options.body = myBody;
            processTestData(options, i);
        }
    });
}

executeTest();

```

Sliku 4.3.1 Komunikacija sa *cloud* servisom glasovne kontrole

4.4 Dobijanje odgovora od *cloud* glasovnog servisa i testiranje

Ceo protokol testiranja obavlja se unutar funkcije *processTestData* (Slika 4.4.1). Kada se dobije odgovor od *cloud* servisa za glasovnu kontrolu, pristupa se poljima koja sadrže

informacije značajne za testiranje, a zatim se učitavaju uređaji iz tekstualne datoteke i vrši se poređenje dobijenog i očekivanog odgovora.

```
function processTestData(options, i) {

    var MAX_TIMEOUT = 4000;
    setTimeout(function() {
        console.log(options.body.request.intent);
        request.post(options, function (error, response, body) {

            if (!error && response.statusCode == 200)

                console.log('Response from Alexa: ' + body.response.outputSpeech.text);

            testResponse = body.response.outputSpeech.text;

            var things = options.body.request.intent.slots.Things.value;
            var requests = options.body.request.intent.slots.DimRequests.value;
            var percentage = options.body.request.intent.slots.Percentage.value;

            var data = fileread("AllDevice.json");

            var array = data.devices;
```

Slika 4.4.1 Funkcija za poređenje očekivanog i dobijenog odgovora

4.4.1 Proces kreiranja očekivanog odgovora

Očekivani odgovor kreiramo u skladu sa poljima unutar poslate JSON datoteke. U nastavku teksta je detaljno opisano kreiranje očekivanog odgovora, za komandu koja testira *Light* servis. Jedna od najbitnijih stvari jeste, da se lista uređaja vezanih za centralni uređaj pribavi na početku testiranja, i smesti u datoteku, kako bi smo izbegli upit pri slanju svake nove komande. Nakon prikupljanja podataka o uređajima, definišu se odgovori koje očekujemo. U konkretnom primeru, kao zahtev *cloud* servisu poslato je paljenje lampe. Odgovore koje možemo da očekujemo su informacije o tome da je lampa upaljena, da je nema u listi uređaja, da se zahtev i uređaj ne podudaraju, te da nismo specificirali uređaj. Kada dobijemo odgovor od *cloud* servisa vršimo pristupanje njegovim bitnim poljima i uzimamo informacije koje su nam potrebne za poređenje u konkretnom primeru uređaj i komadu. Kada se izvrši poređenje očekivanog i dobijenog odgovora vraća se povratna informacija i upisuje se u tekstualnu datoteku za rezultate.

5. Rezultati testiranja

U procesu testiranja neophodno je pokriti različite pozitivne i negativne testne slučajeve. Pozitivni testni slučajevi pokrivaju sve komande u kojima se od sistema očekuje pozitivan odgovor, tj. izvršavanje komande. Negativni testni slučajevi su oni kod kojih se očekuje odgovor o nemogućnosti sistema da izvrši komandu, uz detaljno objašnjenje (fali neki parametar, uređaj ne postoji i slično).

Svaki od testnih slučajeva može biti izvršen uspešno – kada se podudaraju očekivani i stvarni odgovor sistema, ili neuspešno – kada se stvarni odgovor sistema razlikuje od očekivanog. Prilikom testiranja u sistemu pametne kuće postojali su sledeći uređaji:

- 1) Pametna sijalica
- 2) Pametna utičnica
- 3) Prekidači sa potenciometrom
- 4) Senzor za merenje temperature

U tekstu koji sledi su prikazani rezultati testiranja uspešno izvršenih karakterističnih pozitivnih i negativnih testnih slučajeva.

5.1 Rezultati testiranja komadni Dim servisa

Tokom testiranja komandi *Dim servisa*, testirani su sledeći slučajevi :

- Pozitivan test: postavljanje prekidača sa potenciometrom na određenu vrednost.
- Negativan test: odgovor sistema u slučaju da u komandi izostane ime uređaja, željena akcija ili vrednost na koju se prekidač sa potenciometrom postavlja.

U tabeli 5.1.1, su prikazani primeri komandi *Dim servisa* koje su testirane.

Tip slučaja	Opis	Očekivani odgovor
Pozitivni	Postavljanje uređaja sa potenciometrom na određenu vrednost. Komanda je ispravna i sadrži sve neophodne podatke.	Potvrda da je nivo osvetljenja postavljen.
Negativni	Zahtev za postavljanje nivoa osvetljenja izdat nepostojećem uređaju.	Informacija da uređaj ne postoji.
	Zahtev u kome nedostaje željeni nivo osvetljenja.	Informacija da nivo osvetljenja ne postoji.
	Zahtev za postavljanje nivoa osvetljenja iznad ili ispod vrednosti na koju je ograničen uređaj.	Informacija da nivo osvetljenja nije podržan, i da je opseg između 0 i 100.
	Loše formirana komanda, čiji JSON ne sadrži informaciju o željenoj akciji.	Informacija da akcija nije specificirana.

Tabela 5.1.1 Skup komandi Dim servisa

Očekivani rezultati koji se podudaraju sa dobijenim su prikazani na slici 5.1.1.

```
rtrk@rtrkn262-lin:~/Desktop/Test modul za diplomski/DimIntent$ nodejs test.js
Response from Alexa: The dim is set
Expected response: The dim is set
Number of passed test: 1
Number of failure test: 0
Response from Alexa: Percentage value is not in limits, try with value between 0 and 100
Expected response: Percentage value is not in limits, try with value between 0 and 100
Number of passed test: 2
Number of failure test: 0
Response from Alexa: The request is not available for this device.
Expected response: The request is not available for this device.
Number of passed test: 3
Number of failure test: 0
Response from Alexa: This device is not available, try again
Expected response: This device is not available, try again
Number of passed test: 4
Number of failure test: 0
Response from Alexa: There is no percentage provided
Expected response: There is no percentage provided
Number of passed test: 5
Number of failure test: 0
rtrk@rtrkn262-lin:~/Desktop/Test modul za diplomski/DimIntent$ █
```

Slika 5.1.1 Pozitivni i negativni testni slučajevi komandi Dim servisa

5.2 Rezultati testiranja komandi Light servisa

Tokom testiranja komandi *Light* servisa, testirano je sledeće:

- Pozitivan test: paljenje, gašenje i menjanje boja kod pametne sijalice.
- Negativan test: odgovor sistema u slučaju da u komandi izostane ime uređaja, željena akcija ili boja na koju se pametna sijalica postavlja.

U tabeli 5.2.1, su prikazani primeri komandi Light servisa koje su testirane.

Tip slučaja	Opis	Očekivani odgovor
Positivni	Zahtev za paljenje pametne sijalice i pametne utičnice, koji postoje u sistemu.	Informacija da su uređaji upaljeni.
	Zahtev za gašenje pametne sijalice i pametne utičnice, koji postoje u sistemu.	Informacija da su uređaji ugašeni.
	Zahtev za menjanje boje kod pametne sijalice. Uređaj postoji u sistemu, a komanda sadrži sve neophodne podatke.	Informacija da je boja promenjena.
Negativni	Loše formirana komanda, čiji JSON ne sadrži informaciju o željenoj akciji.	Informacija da akcija nije specificirana.
	Zahtev za paljenje uređaja izdat nepostojećem uređaju.	Informacija da uređaj ne postoji.
	Zahtev u kome nedostaje željena boja.	Informacija da boja ne postoji.
	Zahtev izdat uređaju koji ne podržava Light servisu.	Informacija da željena akcija nije podržana za dati uređaj.

Tabela 5.2.1 Skup komandi Light servisa

Rezultati testiranja uspešno izvršenih testnih slučajeva su prikazani na slici 5.2.1.

```
Response from Alexa: The left plug is turned on
Expected response : The left plug is turned on
Number of passed test: 4
Number of failure test: 0
Response from Alexa: The request is not available for this device.
Expected response : The request is not available for this device.
Number of passed test: 5
Number of failure test: 0
Response from Alexa: The left plug is turned off
Expected response : The left plug is turned off
Number of passed test: 6
Number of failure test: 0
Response from Alexa: There is no color provided
Expected response : There is no color provided
Number of passed test: 7
Number of failure test: 0
Response from Alexa: The lamp is changed
Expected response : The lamp is changed
Number of passed test: 8
Number of failure test: 0
Response from Alexa: The lamp is changed
Expected response : The lamp is changed
Number of passed test: 9
Number of failure test: 0
Response from Alexa: The lamp is turned off
Expected response : The lamp is turned off
Number of passed test: 10
Number of failure test: 0
rtrk@rtrkn262-lin:~/Desktop/Test modul za diplomski/LightIntent$
```

Slika 5.2.1 Pozitivni i negativni testni slučajevi komandi Light servisa

Rezultati neuspešno izvršenih testnih slučajeva su prikazani na slici 5.2.2.

```
The lamp is turned on
Number of passed test: 2
Number of failure test: 0
{ name: 'LightIntent',
  slots:
    { LightRequests: { name: 'LightRequests', value: 'turn on' },
      Colors: { name: 'Colors', value: '' },
      FirstLocationDescriptor: { name: 'FirstLocationDescriptor' },
      Things: { name: 'Things', value: 'left plug' },
      RoomDescriptor: { name: 'RoomDescriptor' },
      DeviceEnumeration: { name: 'DeviceEnumeration' },
      SecondLocationDescriptor: { name: 'SecondLocationDescriptor' },
      Floors: { name: 'Floors' },
      Rooms: { name: 'Rooms' },
      FloorDescriptor: { name: 'FloorDescriptor' } } }
The temperature from left plug is 25 degrees
The left plug is turned on
Number of passed test: 2
Number of failure test: 1
```

Slika 5.2.2 Neuspešno izvršen testni slučaj kod Light servisa

5.3 Rezultati testiranja komandi senzor servisa

Prilikom testiranja komandi namenjenih senzorima, pokriveni su sledeći slučajevi:

- Pozitivni test: merenje trenutne temperature.
- Negativni test: odgovor sistema u slučaju da u komandi izostane ime senzora, korišćenje nepostojećeg senzora ili izostavljanje željene akcije.

U tabeli 5.3.1, su prikazani primeri komandi senzor servisa koje su testirane.

Tip slučaja	Opis	Očekivani odgovor
Pozitivni	Zahtev za merenje trenutne temperature. Senzor sa kog se očitava temperatura postoji u sistemu.	Informacija o trenutnoj temperaturi.
Negativni	Loše formirana komanda, čiji JSON ne sadrži informaciju o željenoj akciji.	Informacija da akcija nije specificirana.
	Zahtev za merenje temperature izdat nepostojećem senzoru.	Informacija da senzor ne postoji.
	Izdavanje zahteva za merenje temperature senzoru koji taj servis ne podržava.	Informacija da servis nije podržan.

Tabela 5.3.1 Skup komandi senzor servisa

Rezultati testiranja uspešno izvršenih testnih slučajeva su prikazani na slici 5.3.1.

```
rtrk@rtrkn262-lin:~/Desktop/Test modul za diplomski/SensorIntent$ nodejs test.js
Response from Alexa: This command is not supported for sensors.
Expected response: This command is not supported for sensors.
Number of passed test: 1
Number of failure test: 0
Response from Alexa: The request is not available for this device.
Expected response: The request is not available for this device.
Number of passed test: 2
Number of failure test: 0
Response from Alexa: This device is not available, try again
Expected response: This device is not available, try again
Number of passed test: 3
Number of failure test: 0
Response from Alexa: The temperature from sensor is 27 degrees
Expected response: The temperature from sensor is NUMBER degrees
Number of passed test: 4
Number of failure test: 0
```

Slika 5.3.1 Pozitivni i negativni testni slučajevi komandi namenjenih senzorima

6. Zaključak

Poslednjih godina mogućnost kontrole uređaja glasovnim komandama u pametnim kućama je postala veoma popularna. U cilju omogućavanja glasovnih komandi u okviru postojećeg Oblo sistema kućne automatizacije, izvršena je integracija Oblo sistema sa Amazon Aleksom servisom za prepoznavanje govora. Kreiran je *cloud* modul koji prihvata i izvršava komande koje pristižu sa Aleksa servisa. Prilikom detaljnog testiranja implementiranog modula javila se potreba za razvojem okruženja za automatsko testiranje.

Razvijeno okruženje za testiranje *cloud* sistema glasovne kontrole omogućilo je testiranje na velikom broju testnih slučajeva, kako bi se proverila funkcionalnost *cloud* sistema glasovne kontrole, i utvrdila ispravnost celokupnog sistema pre stavljanja krajnjim korisnicima na raspolaganje. Pored mogućnosti da se izvrši veliki broj testnih slučajeva za kratko vreme, prednost upotrebe automatskog okruženja za testiranje ogleda se i u tome što je omogućeno generisanje i nevalidnih ulaza za implementirani modul, što ne bi bilo moguće kod testiranja izgovaranjem pojedinačnih komandi.

Dalji pravci poboljšanja okruženja za testiranje biće fokusirani na testiranja komandi za pokretanje scena. Okruženje za testiranje je napravljeno tako da se lako može prilagoditi svakoj izmeni u sistemu i omogućiti njen testiranje.

7. Literatura

- [1] Jovan Popović, “Testiranje softvera u praksi”, Beograd , 2012
- [2] ZigBee, [Online] Dostupno na: <http://www.zigbee.org/>
- [3] Z-wave, [Online] Dostupno na: <http://www.z-wave.com>
- [4] OBLO cloud sistem, [Online] Dostupno na: <http://www.obloliving.com/vasa-pametna-kuca/>
- [5] AWS, [Online] Dostupno na: <https://aws.amazon.com/about-aws/>
- [6] Amazon glasovni servis, [Online] Dostupno na: <https://developer.amazon.com/alex-skills-kit>
- [7] JSON format podataka, [Online] Dostupno na: <http://www.json.org/>
- [8] Milan Tucić, “Protokol za razmenu poruka u sistemu pametnih kuća”, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2016
- [9] Node.js, [Online] Dostupno na: <https://nodejs.org/en/>